Physiolibrary - Modelica library for Physiology

Marek Mateják, Tomáš Kulhánek, Jan Šilar, Pavol Privitzer, Filip Ježek, Jiří Kofránek
Institute of Pathological Physiology, 1st Faculty of Medicine, Charles University in Prague
U nemocnice 5, Prague 2, 128 53, Czech Republic
marek@matfyz.cz

Abstract

Physiolibrary is a free open-source Modelica library designed for modeling human physiology. It is accessible on the Modelica Libraries web page at https://www.modelica.org/libraries. This library contains basic physical laws governing human physiology, usable for cardiovascular circulation, metabolic processes, nutrient distribution, thermoregulation, gases transport, electrolyte regulation, water distribution, hormonal regulation and pharmacological regulation.

Keywords: Physiolibrary; HumMod; Modelica library; Physiology; Integrative physiology; System biology

1 Introduction

Our laboratory have a long tradition building physiological libraries, starting with the Matlab/Simulink environment [2]. The origin of this Modelica Physiolibrary was in the first version of our HumMod Golem Edition model implementation [3-6], where it was called HumMod.Library. As the successors of Guyton's Medical Physiology School write, the original HumMod model [7] is "The best, most complete, mathematical model of human physiology ever created" [8].

We are also developing many types of smaller physiological models for use in medical education [9-11], so it was essential to separate this library from our HumMod Modelica implementation. There were already introduced some other Modelica models and libraries in biological domain e.g. [12, 13], which are useful in the process of system modeling and identification or BioChem Modelica library, that implements large part of SBML library in Modelica language [14].

Our Physiolibrary contains only carefully-chosen elementary physiological laws, which are the basis of more complex physiological processes. For example from only three type of blocks (ChemicalReaction, Substance and MolarConservationMass) it is possible to compose the allosteric transitions [15] or the Michaelis-Menten equation.

2 Physiology

Physiology is a very progressive discipline, that examines how the living body works. And it is no surprise that all processes in the human body are driven by physical laws of nature. The great challenge is to marry old empirical experiments with the "new" physical principles. Many teams and projects in the word deal with this formalization of physiology, for example: Physiome[16], SBML[14, 17], EuroPhysiome[18], VPH[19], CellML[20] etc. It is our hope that this library helps this unflagging effort of physiologists to exactly describe the processes.

2.1 Display units in physiology

Energy in medicine and chemistry has a very long tradition. One must not be confused by its different units and definitions. The researcher must be aware of multiple definitions of calorie, such as the international calorie, the 15°C calorie, the thermal calorie or the Calorie with a capital "C". The origin of this unit is in the thermal energy needed to heat one gram of water by one degree Celsius. But because the measurement conditions may differ, these alternative definitions are necessary. In physiology it is recommended to use only international calorie as defined in Table 1. The flow of heat/energy is usually calculated in kcal/min, but in physics this is called power and is expressed in the SI unit watts.

Pressure units in medicine are also mainly based on historical measurements. For many years blood pressure was measured by the mercury sphygmomanometer, where the pressure is represented by the change of mercury hydrostatic column height. And because the scale of units on the column is in millimetres the pressure unit is called millimetre of mercury 'mmHg'. There also exists a very small differ-

ence between this unit and torrs. It is caused again by variance in measurement conditions.

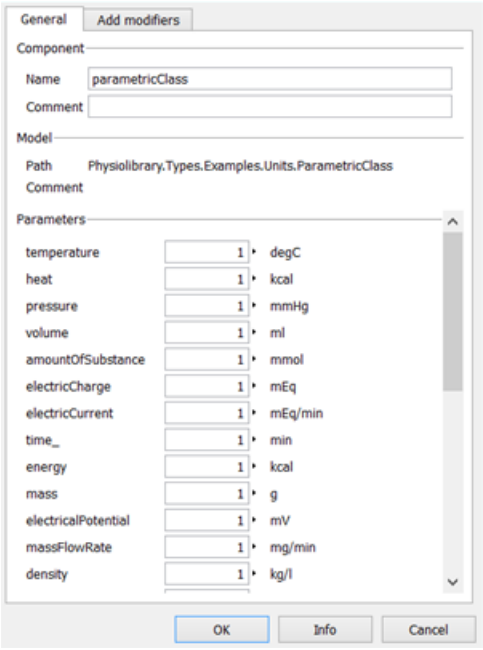

Figure 1, parameter dialog example for non-SI physiological units. Dymola environment automatically converts this user non-SI-values to SI-values in text code to be compatible with any other Modelica libraries.

Many physiological processes are based on electrical principles in the human body. The main cause of this is that each cell has a nonconductive membrane with molecular structures called channels, through which the fluxes of electrolytes can be precisely regulated. Even more, the cells use energy from metabolism to retain a small electric potential between inside and outside. This view leads to a unit called equivalents or "eq". A charge of 1eq, for example, has 1mol of sodium cations (Na^+). The fluxes of electrically charged ions can be in meq/min, but in physics the SI unit ampere is more generally used.

Unit conversion table
(for Modelica environment display-unit setting)

x kcal	=	4186.8*x	J
x kcal/min	=	69.78*x	W
x mmHg	=	133.322387415*x	Pa
x degC	=	273.15 + x	K
x meq	=	96.4853365*x	C
x meq/min	=	1.60808894*x	A
x mosm	=	0.001*x	mol
x litreSTP	=	0.044031617*x	mol
x litreSATP	=	0.040339548*x	mol
x litreNIST	=	0.041571200*x	mol

Table 1, chosen Non-SI units in physiology

Another strange unit describing the amount of substance is the osmol ("osm"), which has the same value as the mol, but which highlights the property that this substance cannot cross the membrane together with the flux of its solvent.

For gases is common to measure amount as volume, which for specific measurement condition has the same meaning as amount of molecules. The International Union of Pure and Applied Chemistry (IUPAC) set this standard condition for temperature and pressure (STP) precisely at 0°C and 100kPa. But also different standards exist. For example (SATP) at 25°C and 100kPa or the standard of the National Institute of Standards and Technology (NIST) at 20°C and 101.325kPa.

2.2 Chemical domains in physiology

In physiology books, chapters about chemical substances are organized by their types. The main reason for this is that each substance in the human body is regulated in a different way. For example the regulation of sodium is different from the regulation of potassium, and from the regulation of glucose, and so on. This view leads to the idea of having separate models of each substance. The origin of different flows and regulations is the (cellular) membrane. Water and solutions can cross it in different directions at the same time. Crossings occur for different reasons: water is driven mostly by osmotic gradients, electrolytes are driven by charge to reach Donnan's equilibrium, and some solutes can even be actively transported against their concentration or electrical gradients. And all this is specifically driven from the higher levels by neural and hormonal responses.

In Physiolibrary flows and fluxes of solutes are supported mostly by the Chemical package. All parts inside this Physiolibrary.Chemical package use the connector ChemicalPort, which defines the molar concentration and molar flow/flux rate of one solute. This is the supporting infrastructure for modeling membrane diffusion, accumulations of substances, reversal chemical reactions, Henry's law of gas solubility, dilution with additional solvent flow, membrane reabsorption, chemical degradation and physiological clearance.

For usage examples, please open the Chemical.Examples package, where is implemented chemical reaction as shown in Fig. 2 or many other chemical processes.

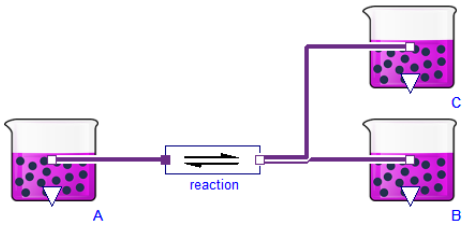

Figure 2, example of chemical reaction A <-> B + C. Purple beakers (Substance) accumulate one type of substance and generate its concentration at port. Block for chemical reaction (ChemicalReaction) is designed to any number of substrates or products with any stoichiometric numbers. In this case there are only one substrate and two products. Purple lines represent the connection of chemical connectors, which is composed with molar concentration and molar flow of substance.

Graphically user created and set diagram of example in Fig. 2 generate this Modelica code:

```
model SimpleReaction2
  import Physiolibrary.Chemical.Components.*;
  Substance           A(solute_start=0.9);
  ChemicalReaction    reaction(K=1, nP=2);
  Substance           B(solute_start=0.1);
  Substance           C(solute_start=0.1);
equation
  connect(A.q_out, reaction.substrates[1]);
  connect(reaction.products[1], B.q_out);
  connect(reaction.products[2], C.q_out);
end SimpleReaction2;
```

where before the numerical simulation each parameter Substance.solute_start must be set to some initial amount of substance, ChemicalReaction.nP to the number of products and parameter ChemicalReaction.K to the dissociation constant of reaction in SI-units (please notice, that concentration of 1 mol/m^3 = 1 mmol/L). As has written before the values in text code are in SI-units, but Dymola environment support non-SI units in parameter dialog of each component.

2.3 Hydraulic domain in physiology

The main usage of the hydraulic domain in human physiology is modeling of the cardio-vascular system. And because there are no extreme thermodynamic conditions, the system can be really simple —it is only necessary to model conditions for incompressible water, at normal liquid-water temperatures and with relative pressure 5-20kPa. This boring thermodynamic state leads to the very simple blocks of hydraulic resistance, hydrostatic pressure, volumetric flow, inertia and finally the block of blood accumulation in elastic vessels.

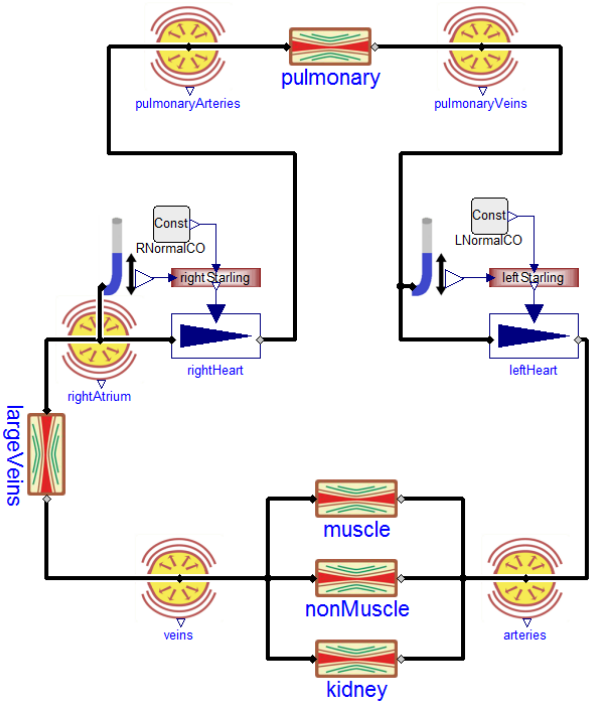

Figure 3, hydraulic example: cardiovascular subsystem of famous Guyton-Coleman-Granger model [1]. Yellow circles (ElasticVessel) represent blood accumulation and pressure generation, rectangles between them are hydraulic resistances (Resistance) of blood vessels, blue triangles (Pump) represent heart pump driven by Frank-Starling law. Heart filling pressures are determined by block with icon of blue tube (PressureMeassure) and block-rectangle (Blocks.Factors.Spline) convert filling pressure to effect on cardiac output. Black lines are connecting the hydraulic connectors (PressureFlow), which contains pressure and volumetric flow variables.

2.4 Thermal domain in physiology

For the human body to function optimally, it is critical to hold the core temperature at 35–39°C. A fever of 41°C for more than a short period of time causes brain damage. If the core temperature falls below 10°C, the heart stops. As in the hydraulic domain, the thermal domain is simplified to these conditions.

In the Physiolibrary.Thermal package extends the package Modelica.Thermal.HeatTransfer from Modelica Standard Library 3.2 (MSL), where the connector is composed of temperature and heat flow. The main blocks in Physiolibrary.Thermal are: Conductor, IdealRadiator and HeatAccumulation. The heat conductor conducts the heat from the source, such us muscles or metabolically active tissue, to its surrounding. IdealRadiator delivers heat to tissues by blood circulation. HeatAccumulation plays a role in

accumulating thermal energy in each tissue mass driven by its heat capacity. We recommend to use this block instead of Modelica.Thermal.HeatTransfer.HeatCapacitor to have possibility of variable mass amount or to have a support for calculation of steady state, described in section 2.7.

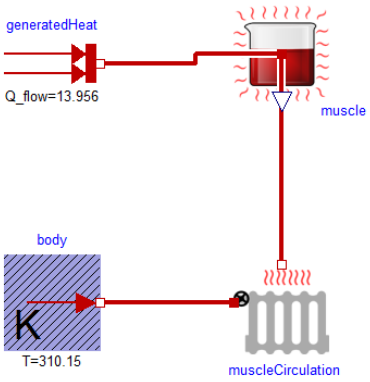

Figure 4, example of heat flow from working muscle. Muscle is represented by red beaker (HeatAccumulation), where heat energy is accumulated in mass with defined weight and specific heat. Heat transfer is processed by blood circulation (IdealRadiator) with blood flow as its internal parameter. The temperature of blood is set to fixed value of 37°C to simulate well regulated core body temperature.

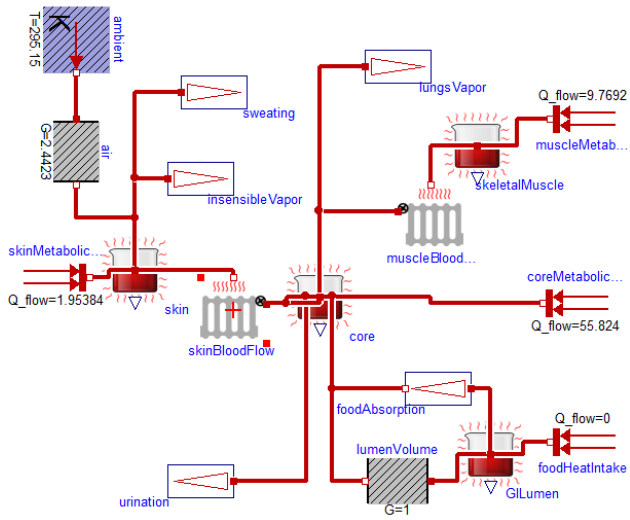

Figure 5, basic heat flow model of human body. Heat production is done in each tissue by metabolism or from food using MSL block with two red arrows icon (Modelica.Thermal.HeatTransfer.Sources.FixedHeatFlow), heat is stored in tissues (HeatAccumulation) and transferred by blood (IdealRiator) or with together with mass (Stream, HeatOutstream), where can be integrated also vaporization heat loss. Heat radiation and conduction to environment is simplified using MSL block for heat conductor (Modelica.Thermal.HeatTransfer.Components.ThermalConductor).

2.5 Osmotic domain in physiology

One of the basic phenomenon of biological systems is the osmotically-driven flow of water. This is always connected with semipermeable membranes. The different concentrations of impermeable solutes on both sides of the membrane causes the hydrostatic pressure at the concentrated side to rise[21]. This pressure difference is called osmotic pressure. Osmotic pressure is linearly proportional to the concentration gradient of impermeable solutes. The osmolarity (osmotic concentration) is also one of the main indexes of human body balance, called homeostasis. Its value should not significantly deviate for a long period of time from a value of 285-295 mosm/l.

In Physiolibrary the osmotic connector OsmoticPort is composed of the osmotic concentration and the volumetric flux of permeable liquid. The two main blocks are called Membrane and OsmoticCell. Here, inside the membrane blocks, it is of course possible to also define hydraulic pressure and temperatures effects on both sides of membrane.

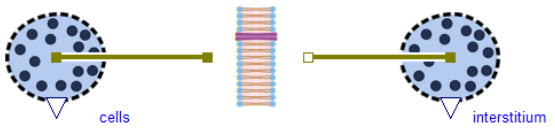

Figure 6, osmotic example to simulate water transfer between intracellular and interstitial compartment in hypertonic or hypotonic conditions.

2.6 Types and units

The most common errors in HumMod Golem Edition were caused by using bad physical units. The main problem of medical research, articles, and experiments is using obscure units from medicine, pharmacology, biology and non-physics disciplines.

The Physiolibrary fulfills the Modelica ideal of using SI units as the main unit for each variable, and the previously described physiological units are also implemented as the displayUnits for each variable. Using these displayUnits the user sets and sees the "physiological" values. The implementation can also be joined to any unit-correct Modelica models and physical equations without crashing due to unit incompatibilities. The unit support of Physiolibrary is so strong that one even can chose the right unit-typed “input real”/”output real” from the library package Types.RealIO or one can use unit-typed constant (Types.Constants). As can be expected, only the non-

specific package Blocks in the Physiolibrary has variables without units.

2.7 Steady states

One of the main question in clinical medicine is how to stabilize the patient. In the fact of the oscillating heart, breathing, circadian rhythm or menstruation cycle the model could be designed as non-oscillating with variables such as period times, amplitudes, frequencies, mean values and other phase space variables. This type of model has better numerical stability for longer simulation time and even more it can be "stabilized". This stabilization we called steady state.

To be mathematically exact, we define an ***steady state system*** (SSS) as a non-differential system derived from a original differential system (DS) by using zero derivations and by adding additional steady state equations (ASSE). The number of the ASSE must be the same as the number of algebraically dependent equations in the non-differential system derived from DS by setting zero derivations. The ASSE describes the system from the top view mostly such as the equations of mass conservation laws or the boundary equation of environment sources.

To define a model as an SSS the user must switch each Simulation parameter in each block to value Types.SimulationType.SteadyState and must have correctly defined all necessary ASSE. This setting caused to ignoring any start values for any state and add zero derivation equations instead. Today does not exist Modelica environment, which could automatically find and remove generated dependent equations by this way. So the correct number of states must be marked as dependent (parameter isDependent) and the same number of ASSE must be inserted. Despite the fact, that model in this steady-state setting will be not locally balanced it should be globally balanced and without any dependent equation.

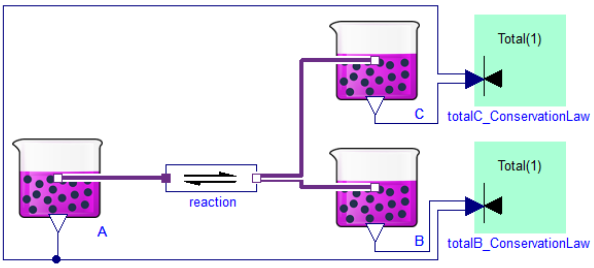

Figure 7, steady state system example: equilibrium of chemical reaction A<->B+C is calculated with two ASSE. Zero change of reactant A is automatically propagated trough reaction to both products. So both products must be marked as dependent (parameter isDependent) and two mass conservation laws must be added as green square blocks in the right (MolarConservationLaw). Please note, that conservation laws must be included after designing of the whole system, because they are global properties, not only properties of separated substances or reactions.

Adding of one ASSE is possible by inserting and connecting of the energy or mass conservation law block from package SteadyState.Components. Other possibilities is in blocks of environment sources, where the setting of parameter isIsolatedInSteadyState add the equation of the zero mass/volume/energy flow from or to environment.

The model in steady state often changes to one big nonlinear strong component, but without solver stiff or convergence problems. Especially in quick chemical reaction kinetics is not necessary to have very rapid molar fluxes, when it always reach equilibrium. This design also approve to create steady stated parts in dynamical model without huge rebuilding. It also brings other benefits. To see these possibilities, one have to realize that conservation laws could be invariances in a dynamical simulation. This is really useful for debugging.

For example see the model SteadyStates.Examples.SimpleReaction2_in_Equilibrium (Fig.7), which implements the equilibrium of the closed system from Fig.2 as a solution of three chemical substances with a simple reversible reaction between them extended by two conservation laws. Each of this law describe total possible amount of one product in its free form and in its asociated form.

It is always a big challenge to nicely solve initial values of differential system. However, it should be possible to solve the SSS in initial phase. And this is the idea behind the Types.SimulationTypes.InitSteadyState option for models already extended with ASSE to support SSS.

2.8 File utilities—input/output manipulations

During the creation and debugging of huge integrated models it is necessary to easily define consistent input, output and test sets of all output variables for some subsystems. Let's imagine that we have a model composed only of subsystems that converge from some constant inputs to constant outputs. It should be possible to substitute each main subsystem for its chosen constant output values as parameters. Comparing the model with these parametric values and the original subsystem can show the wrong part of the simulation.

For example in the huge HumMod model it is necessary to debug smaller parts separately. These

tools could be use, because HumMod is the type of constant-converged model. Each subsystem in the first level has the constant input values set for its output variables. Simulating, for example, the cardiovascular subsystem is possible by creating the high-level system with the original cardiovascular subsystem, but with a constant metabolic, constant thermoregulation, constant hormonal, constant water, constant proteins, constant gases, constant electrolytes and constant status subsystem.

Because the number of output variables for each subsytem changes during development, it is a good idea to have only one list for each subsystem. And generating consistent sets to store, restore, compare initial and final values is possible by the same pattern as presented in the package Types.Example. In this package it is also possible to define a customized way to save and load the variables that connect subsystems together. For this purpose, one has to redeclare the package Types.Utilities with simple functions for reading and writing values, such as is done in the default package FileUtilities.

The typical code of parameter set could be:

```
model MyParameterSet
  replaceable package T = Physiolibrary.Types.RealTypes
      constrainedby Physiolibrary.Types.RealTypes;

  T.Pressure Bone_PO2(
                          varName="Bone-Flow.PO2");
  T.VolumeFlowRate BoneBloodFlow(
                          varName="Bone-Flow.BloodFlow");
  T.MolarFlowRate BoneO2Need(
                          varName="Bone-Metabolism.O2-Need");
  T.Volume BoneLiquidVol(
                          varName="Bone-Tissue.LiquidVol");

  BusConnector busConnector;
equation
  connect(Bone_PO2.y, busConnector.Bone_PO2);
  connect(BoneBloodFlow.y, busConnector.Bone_BloodFlow);
  connect(BoneO2Need.y, busConnector.Bone_O2Need);
  connect(BoneLiquidVol.y, busConnector.Bone_LiquidVol);
end MyParameterSet;
```

To redefine these defined set of values to inputs from file is possible just by redeclaring type of values to type InputParameter and redirecting reading functions to FileUtilities:

```
model InputParameterSet
extends MyParameterSet( T(redeclare block Variable =
     Physiolibrary.Types.RealExtension.InputParameter (
       redecla-
re package Utilities = Physiolibrary.Types.FileUtilities)));
end InputParameterSet;
```

And the same set of values can be redefined also to file output in the end of simulation:

```
model OutputFinalSet
extends MyParameterSet( T(redeclare block Variable =
     Physiolibrary.Types.RealExtension.OutputFinal (
       redecla-
re package Utilities = Physiolibrary.Types.FileUtilities)));
end OutputFinalSet;
```

3 Conclusion

In our opinion the best way to understand this library is to download it from the Modelica web pages at www.modelica.org/libraries and examine the examples. We recommend examining the package Hydraulica.Examples, which provides an example of a simplified cardiovascular system; the package Chemical.Examples, which provides an example of allosteric hemoglobin oxygen binding; the package Osmotic.Examples, which simulates cell volume in hypertonic and hypotonic environments; and finally the package Thermal.Examples, which simulates the heating of circulated blood inside active muscles.

Acknowledgements

This paper describes the outcome of research that has been accomplished as part of a research program funded by the Ministry of Industry and Trade of the Czech Republic by the grant FR—TI3/869 and by The Ministry of Education, Youth and Sports by the grant SVV-2013-266509.

In addition the main author wants to thank Jan Obdržálek for consultations about physical units and Austin David Schaefer for corrections of English language.

References

1. Guyton, A.C., T.G. Coleman, and H.J. Granger, *Circulation: overall regulation.* Annual review of physiology, 1972. **34**(1): p. 13-44.
2. Teaching, L.o.B.a.C.A. *Physiolibrary in Matlab and Simuling*. 2008; Available from: http://www.physiome.cz/simchips.
3. Marek Mateják and J. Kofránek, *HUMMOD–GOLEM EDITION–ROZSÁHLÝ MODEL FYZIOLOGICKÝCH SYSTÉMŮ* Medsoft, 2011: p. 182-196.
4. Marek Mateják and J. Kofránek, *Rozsáhlý model fyziologických regulací v Modelice.* Medsoft, 2010: p. 126-146.

5. Kofránek, J., Mateják,Marek, Privitzer,Pavol. *HumMod - large scale physiological model in Modelica*. in *8th. International Modelica conference*. 2011. Dresden, Germany.
6. Kofránek, J., Mateják, M., Privitzer, P., Tribula, M., Kulhánek, T., Šilar, J., Pecinovský, R. *HumMod-Golem Edition: large scale model of integrative physiology for virtual patient simulators*. in *World Congress in Computer Science 2013 (WORLDCOMP'13), International Conference on Modeling, Simulation and Visualisation Methods (MSV'13)*. 2013.
7. Hester, R.L., et al., *HumMod: a modeling environment for the simulation of integrative human physiology.* Frontiers in physiology, 2011. **2**.
8. Center, L.b.U.o.M.M., *HumMod.* 2012: p. http://hummod.org/.
9. Kofránek, J., M. Mateják, and P. Privitzer. *Leaving toil to machines - building simulation kernel of educational software in modern software environments*. in *Mefanet 2009*. 2009. Masaryk University, Brno.
10. Kofránek, J., M. Mateják, and P. Privitzer, *Web simulator creation technology.* MEFANET report, 2010. **3**: p. 52-97.
11. Kofranek, J., et al., *The Atlas of Physiology and Pathophysiology: Web-based multimedia enabled interactive simulations.* Computer methods and programs in biomedicine, 2011. **104**(2): p. 143-153.
12. Proß, S. and B. Bachmann, *An Advanced Environment for Hybrid Modeling and Parameter Identification of Biological Systems.*
13. Cellier, F.E. and A. Nebot. *Object-oriented Modeling in the Service of Medicine*. in *Proc. 6th Asia Simulation Conference*. 2005.
14. Hucka, M., et al., *The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models.* Bioinformatics, 2003. **19**(4): p. 524-531.
15. Monod, J., J. Wyman, and J.-P. Changeux, *On the nature of allosteric transitions: a plausible model.* Journal of Molecular Biology, 1965. **12**(1): p. 88-118.
16. Bassingthwaighte, J.B., *Strategies for the physiome project.* Annals of Biomedical Engineering, 2000. **28**(8): p. 1043-1058.
17. Brugård, J., et al. *Creating a Bridge between Modelica and the Systems Biology Community*. in *7th International Modelica Conference, Como, Italy*. 2009.
18. Fenner, J.W., et al., *The EuroPhysiome, STEP and a roadmap for the virtual physiological human.* Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2008. **366**(1878): p. 2979-2999.
19. Hunter, P.J. and M. Viceconti, *The VPH-physiome project: standards and tools for multiscale modeling in clinical applications.* Biomedical Engineering, IEEE Reviews in, 2009. **2**: p. 40-53.
20. Smith, L., et al., *SBML and CellML translation in Antimony and JSim.* Bioinformatics, 2013: p. btt641.
21. Mortimer, R.G., *Mathematics for physical chemistry*. 1999: Access Online via Elsevier.